Bright Cluster Manager 8.1

Machine Learning Manual

Revision: 7bbb6d4

Date: Wed Dec 12 2018

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. PBS Professional, PBS Pro, and Green Provisioning are trademarks of Altair Engineering, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *Machine Learning Manual* for Bright Cluster Manager 8.1.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage basic machine learning capabilities easily using Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 8.1 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Big Data Deployment Manual* describes how to deploy Big Data with Bright Cluster Manager.
- The *Machine Learning Manual*—this manual—describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

There is also a feedback form available via Bright View, via the Account icon,  , following the clickpath:

`Account`→`Help`→`Feedback`

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website `https://support.brightcomputing.com`. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 13.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Introduction And Installing the Machine Learning RPMs

1.1 Introduction

From Bright Cluster Manager version 7.3 onwards, a number of machine learning and deep learning library and framework RPM packages can be used. The packages provided make it faster and easier for organizations to install the latest state-of-the-art libraries, and to gain insights from rich, complex data.

From Bright Cluster Manager version 8.1 onwards, the machine learning and deep learning packages are normally only accessible with the data science add-on enabled.

- The add-on is typically sold by a Bright Computing sales representative along with the main cluster manager product. The installation ISO is then made with the data science add-on pre-enabled in the cluster configuration settings. So, when the installation is carried out with this ISO, the data science add-on is in an enabled state on the cluster once the cluster is up. This situation is described in section 1.1.1.
- If the add-on is not sold at the same time, but is sold some time after the installation ISO has been made, then, if the cluster installation was already carried out from that ISO, then the data science add-on on the cluster is not in an enabled state on that cluster. How to enable the data science add-on in this situation is described in section 1.1.2.

1.1.1 Data Science Add-On Pre-Enabled In Bright Cluster Manager ISO

The data science add-on is already pre-enabled for an ISO downloaded with the product license, from Bright Cluster Manager version 8.1 onwards, only if the sales representative has included that option in the license.

If the data science add-on is not pre-enabled for a cluster in its license, it means that the cluster ISO does not have the add-on, and so when the cluster is up and running, it does not have the add-on enabled. In this case, the add-on can be enabled by following the procedure described in section 1.1.2.

If CMDaemon is running, then the output of the `licenceinfo` command indicates if the data science add-on is in the enabled state. If the value of `Max Data Science Nodes` in the output is greater than 0, then the data science add-on is in the enabled state. For example (some output elided):

Example

```
[root@bright81 ~]# cmsh -c "main; licenseinfo"
License Information
---------------------------- ----------------------------------
...
Version                      7.0 and above
Edition                      Advanced
```

```
Pre-paid Nodes              100
Max Pay-per-use Nodes       1000
Max Data Science Nodes      80
Max OpenStack Nodes         70
Node Count                  4
Accounting & Reporting      Yes
...
```

The data science add-on is activated and functional when the product license is activated and installed (Chapter 4 of the *Installation Manual*).

1.1.2 Data Science Add-On Not Pre-Enabled In Bright Cluster Manager Installation ISO

An existing cluster may be installed without the data science add-on in an enabled state, and then be relicensed for the data science add-on. To bring the data-science add-on to an enabled state, the administrator should use the wizard at `http://licensing.brightcomputing.com/licensing/activate-data-science/`. This enables and activates the new license so that the data science add-on can function on that existing cluster.

1.2 Packages Available

Currently the following RPMs are available when the data science add-on is enabled and activated.

Table 1.1: *Machine Learning packages provided by the Bright Cluster Manager repositories*

Package name	Description
`cm-ml-distdeps`	Meta-package containing library dependencies for Caffe, NVIDIA DIGITS, Theano, Torch, and TensorFlow. This should be installed on the head nodes and the software images
`cm-ml-pythondeps`	Pre-packaged Python dependencies for Bright's RPM packages for Caffe, NVIDIA DIGITS, Theano, Torch, and TensorFlow
`caffe`	The version 2 branch of the lightweight, modular, and scalable Caffe deep learning framework. With MPI support.
`cudnn`	CUDA deep neural network primitives library
`torch7`	A Lua(JIT)/C++ library for developing Open Source speech and machine learning applications.
`pytorch`	Python 2.7 Torch package, with MPI support
`pytorch-python3`	Python 3 Torch package, which includes torchvision, Caffe2, and MPI support
`theano`	A Python library that allows mathematical expressions involving multi-dimensional arrays to be defined, optimized, and evaluated efficiently.

...continues

Table 1.1: Machine Learning Packages Included...continued

Package name	Description
tensorflow	TensorFlow is an Open Source Software Library for Machine Intelligence
tensorflow-python3	Python 3 TensorFlow package
horovod	Deep learning library that uses MPI, used for Theano and TensorFlow.
keras	Meta-framework deep learning library, used to provide a standard interface for Theano, TensorFlow, cntk, pytorch, mxnet.
keras-python3	Python 3 Keras package
cub	Reusable software components for CUDA
nccl	(pronounced "Nickel") NVIDIA Collective Communication Library — A standalone CUDA library similar in concept to MPI. It has communication routines that have been optimized to achieve high bandwidth over PCIe and other interconnects
mlpython	MLPython is a library for organizing machine learning research.
digits	DIGITS is a web frontend to Caffe and Torch, developed by NVIDIA
cm-jupyter	Jupyter Notebook is a BSD-licensed web notebook
cm-jupyterhub	JupyterHub is a multi-user server for notebooks
cntk	The Cognitive Toolkit by Microsoft Research, is a unified deep-learning toolkit
mxnet	A flexible, compact, and highly scalable Deep Learning library.
mxnet-python3	Python 3 MXNet package

The following packages are planned for the future:

Table 1.2: *Machine Learning packages planned*

Package name	Description
caffeonspark	CaffeOnSpark brings deep learning to Hadoop and Spark clusters. By combining salient features from deep learning framework Caffe and big-data frameworks Apache Spark and Apache Hadoop, CaffeOnSpark enables distributed deep learning on a cluster of GPU and CPU servers.
bidmach	BIDMach is a very fast machine learning library.

1.3 Requirements

The folowing requirements must be met before installing the preceding machine learning packages

- The base distribution used must be RHEL, Centos or Scientific Linux 7.x
- There must be access to the Linux distribution's online YUM repositories as well the EPEL repository
- There must be enough free space for the RPMs that are installed on the head node and compute nodes. The amount depends on the packages installed. 2GB is a minimum and is unlikely to be enough. Installing Digits takes up about 10GB. An additional 400 MB is needed at least for each software image for running machine learning pipelines

- 8GB of RAM on the nodes is the minimum recommended amount.
- The NVIDIA GPUs must be be Maxwell or more recent, with compute capability 3.5 or later. CUDA 6.0 is recommended.
- The CPU must support the AVX/AVX2, FMA, SSE4.2 instructions. This can be checked by inspecting the CPU flags:

  Example

```
[root@node ~]# egrep -m1 -o '(avx|avx2|fma|sse4_2)' /proc/cpuinfo
fma
sse4_2
avx
avx2
```

1.3.1 Software Installation

Compute Nodes Installation

The `cm-ml-distdeps` RPM package must be installed onto all compute nodes that are to run machine learning applications. The `cm-ml-distdeps` meta-package instructs YUM to install the necessary system libraries as well as the development packages, e.g. `blas-devel`.

For example, if the name of the software image is `gpu-image`, then the administrator can install the RPM as follows:

Example

```
[root@bright81 ~]# yum install --installroot=/cm/images/gpu-image cm-ml-distdeps
```

The preceding command must be applied to all software images that are used to run machine learning applications.

Head Node Installation

The head node must also have the `cm-ml-distdeps` RPM package installed on it. If it is not already installed, then it should be installed:

Example

```
[root@bright81 ~]# yum install cm-ml-distdeps
```

The Bright Cluster Manager machine learning packages have proper RPM dependencies defined. This means that the cluster administrator does not need to spend time figuring out what needs to be installed.

For example, if the administrator wants to install NVIDIA's DIGITS, then all that needs to be done is to run the following command on the head node:

```
[root@bright81 ~]# yum install digits
```

In general, a `yum install` *<name of desired package>* should do installation for supported machine learning packages. That is, YUM then automatically installs `cm-ml-pythondeps`, `cm-ml-distdeps`, `cudnn`, `caffe`, `torch` as dependencies if needed.

The RPMs get installed in the `/cm/shared` directory, which is exported over NFS. The RPMS are therefore available to all the compute nodes. The installation of the machine learning libraries is therefore done within minutes, rather than the days that it typically takes to build and install all the necessary dependencies.

Software Libraries Useful For Developers

Developers that work on extending the machine learning libraries typically do not want to use the pre-packaged RPMs. For this use case, Bright can help minimize the time spent to get started.

By installing the `cm-ml-distdeps` on the head node and compute nodes, and the `cm-ml-pythondeps` and `cudnn` RPM packages on the head node, a developer can get ready for machine learning development within minutes and spend time on the interesting application at hand, rather than wasting time in dependency hell.

1.3.2 Module Installation

Bright provides environment module definitions for all the machine learning packages. The environment module files are also compatible with the Lmod software introduced in Bright Cluster Manager 7.3.

The machine learning environment modules automatically load additional environment modules as dependencies.

For example, loading the DIGITS module with:

```
module load digits
```

automatically loads additional modules such `cudnn`, `openblas`, `hdf5_18`, and so on. Those modules are the dependencies needed to use DIGITS.

Example

```
[root@bright81 ~]# module list
Currently Loaded Modulefiles:
  1) shared       2) cmsh       3) cmd      4) cluster-tools/8.1
[root@bright81 ~]# module load digits
[root@bright81 ~]# module list
Currently Loaded Modulefiles:
 1) shared                   7) openblas/dynamic/0.2.20  13) gcc5/5.5.0
 2) cmsh                     8) cuda91/toolkit/9.1.85    14) caffe/0.16.5
 3) cmd                      9) hdf5_18/1.8.20           15) torch7/7.0
 4) cluster-tools/8.1       10) nccl2/2.1.2              16) tensorflow/1.5.0
 5) cm-ml-pythondeps/2.0.0  11) protobuf/3.2.1           17) digits/6.1.1
 6) cudnn/7.0               12) opencv3/3.1.0
[root@bright81 ~]#
```

The module dependencies are achieved via the module definition files:

```
[root@bright81 ~]# module show digits
-------------------------------------------------------------------
/cm/shared/modulefiles/digits/6.1.1:

module-whatis   adds nVidia Deep Neural Network Library to your environment variables
module load     caffe
module load     torch7
module load     cm-ml-pythondeps
module load     openblas
module load     cuda91/toolkit
module load     hdf5_18
module load     tensorflow
prepend-path    PATH    /cm/shared/apps/digits/6.1.1/
prepend-path    PYTHONPATH      /cm/shared/apps/digits/6.1.1
-------------------------------------------------------------------
```

1.3.3 Further Reading

Additional information about the usage of each individual framework may be found in the user manual.

2

Running Caffe

This chapter goes through an example workflow in order to train a Caffe model to recognize handwritten digits. It closely follows the sample run by NVIDIA's Luke Yeager at `https://github.com/NVIDIA/DIGITS/blob/master/docs/GettingStarted.md`. NVIDIA GPUs are required for the version of Caffe used in the current release.

The example uses the MNIST handwritten digit database (`http://yann.lecun.com/exdb/mnist`) as the input to provide a training and validation dataset, and LeNet-5 (`http://yann.lecun.com/exdb/lenet/`) for the neural network model that is to be trained to classify the dataset. Both are generously made available by Yann LeCun from his website at `http://yann.lecun.com/`.

2.1 Downloading The MNIST Data

The MNIST dataset can be downloaded using the DIGITS downloader. For example, a user `tim`, could unpack the datasets into a directory `/home/tim/` as follows):

```
[tim@bright81 ~]$ python /cm/shared/apps/digits/current/digits/download_data/__main__.py \
 mnist /home/tim/
Downloading url=http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz ...
Downloading url=http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz ...
Uncompressing file=train-images-idx3-ubyte.gz ...
Uncompressing file=train-labels-idx1-ubyte.gz ...
Uncompressing file=t10k-images-idx3-ubyte.gz ...
Uncompressing file=t10k-labels-idx1-ubyte.gz ...
Reading labels from /home/tim/train-labels.bin ...
Reading images from /home/tim/train-images.bin ...
Reading labels from /home/tim/test-labels.bin ...
Reading images from /home/tim/test-images.bin ...
Dataset directory is created successfully at '/home/tim/'
Done after 40.9197380543 seconds.
[tim@bright81 ~]$
```

2.2 Setting The Default Matplotlib Backend

The administrator can set the default `matplotlib` GUI backend supported by editing the file:

```
/cm/shared/apps/cm-ml-pythondeps/lib64/python2.7/site-packages/matplotlib/mpl-data/
matplotlibrc
```

and, for example, changing the line:

```
backend: agg
```

to:

```
backend: gtk3agg
```

The preceding change allows GTK3 GUI support, if, for example, just basic X11 GUI support is not wanted, and GTK3 is installed.

2.3 Creating A Working Directory

There must be a working directory path for DIGITS. Its value is defined by the parameter `jobs_dir` in:

```
/cm/shared/apps/digits/current/digits/digits.cfg
```

The value is set to `/tmp/digits/jobs` by default. However, the directory is not yet created. The user must therefore run a command to create the directory, such as:

```
[tim@bright81 ~]$ mkdir -p /tmp/digits/jobs
```

The administrator can define and create a less public working directory according to preference.

2.4 Using The Web App

The DIGITS server can be started with:

```
[root@bright81 ~]# module load shared digits
[root@bright81 ~]# cd /cm/shared/apps/digits/current
[root@bright81 current]# ./digits-devserver

  ___ ___ ___ ___ ______ ___
 |   \_ _/ __|_ _|_   _/ __|
 | |) | | (_ || |  | | \__ \
 |___/___\___|___| |_| |___/ 6.1.1

2017-08-11 08:33:45 [INFO ] Loaded 4 jobs.
```

Caffe requires an NVIDIA GPU with compute capability 3.5 or higher, which means CUDA version 3.5 or higher. CUDA 6.0 or higher is recommended. If there is a GPU on the node, and an error response is displayed, then a possible cause is that the NVIDIA CUDA driver has not been loaded, or has loaded incorrectly. NVIDIA CUDA installation is described in section 7.5 of the *Installation Manual*.

Once the DIGITS server is up and running, a web browser can be used to navigate to the home screen of DIGITS. If all is well, then the server location should be indicated by the last line. The locations are typically at:

- `http://localhost/`

  or at

- `http://localhost:5000/` if using digits-devserver

  or at

- `http://localhost:34448/` if using digits-server

Instead of using `localhost` in the preceding URLs, the external IP address of the system can be used, if the Shorewall firewall is opened for the appropriate port. Typically the administrator adds lines to the `rules` file, similar to the output of the following grep:

```
[root@bright81 ~]# grep 5000 /etc/shorewall/rules
# -- Allow port 5000 (digits)  traffic from elsewhere to master
ACCEPT:info   net             fw              tcp     5000  #DIGITS
```

After adding the lines, Shorewall should be restarted, with, for example, `service shorewall restart`.

The DIGITS home screen should then show up in the browser (figure 2.1).

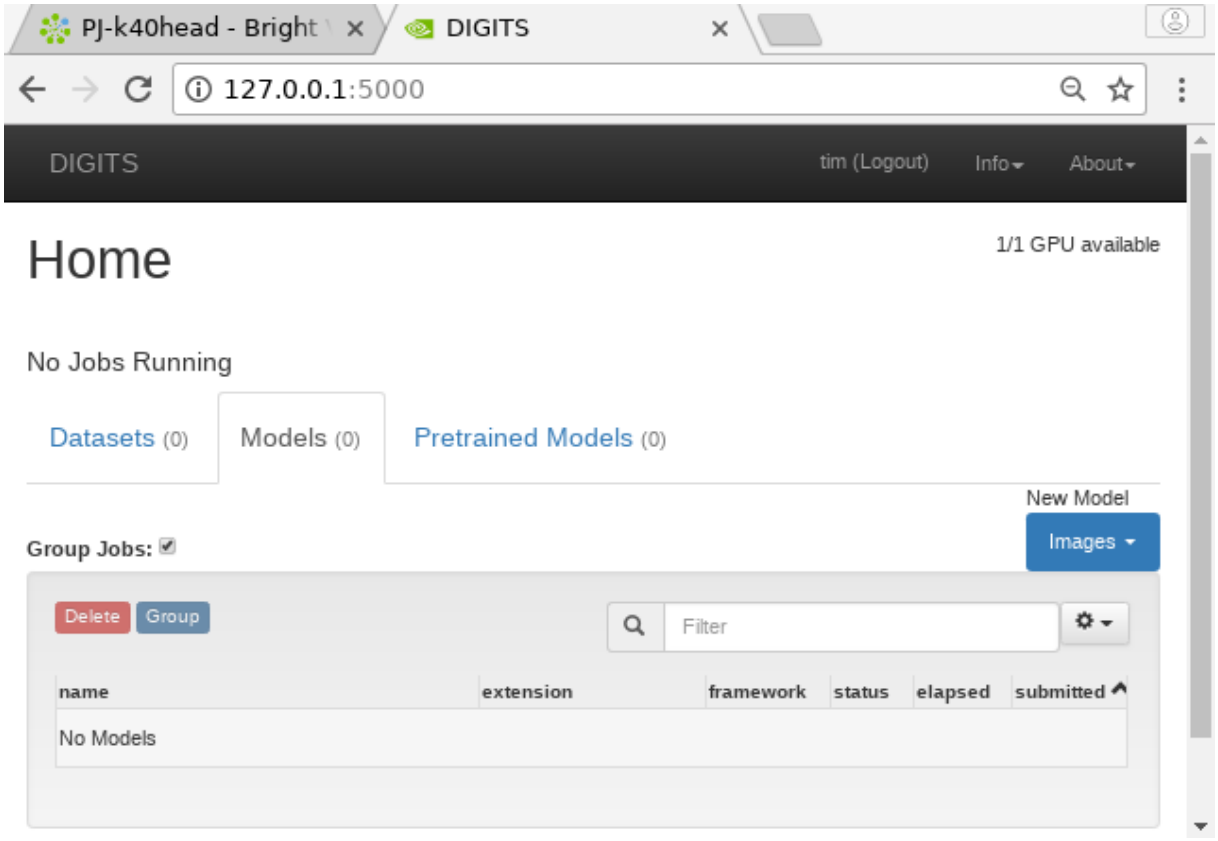

Figure 2.1: DIGITS Home Screen

2.5 Logging In

Clicking on the `Datasets` tab, then on the `Images` menu button under the `New Dataset` label, opens up a menu. Choosing the `Classification` menu item leads to the login page (figure 2.2). The login is carried out without authentication, for convenience. It is not a security feature.

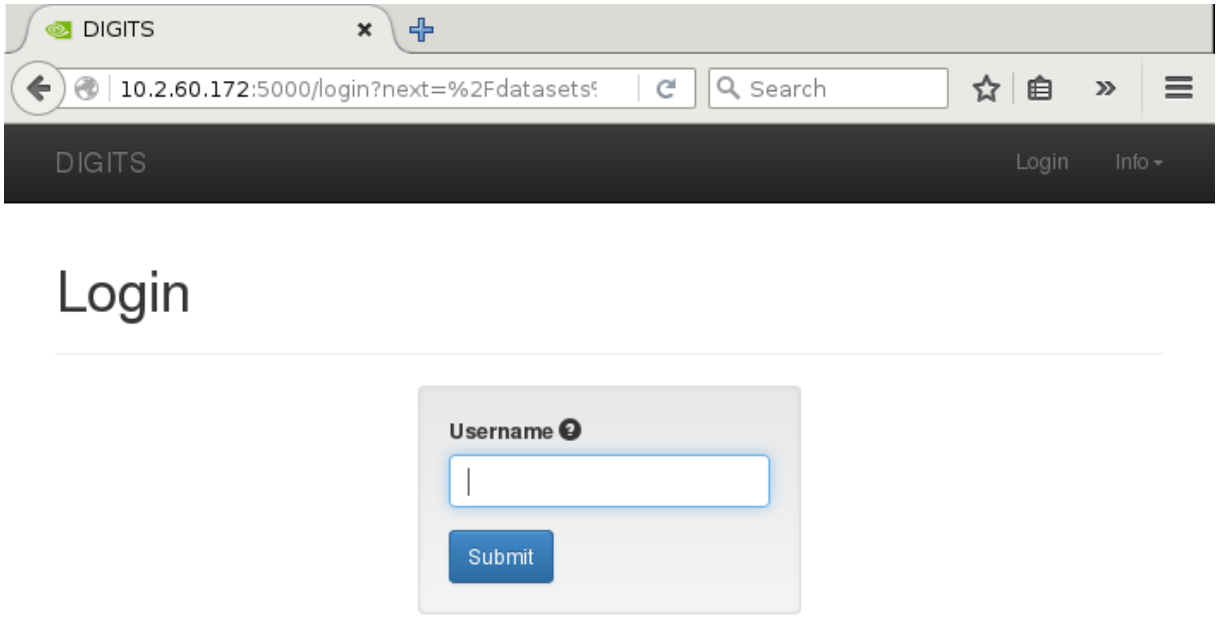

Figure 2.2: DIGITS Login

2.6 Creating Training And Validation Datasets

After login the `New Image Classification Dataset` page appears (figure 2.3). The page shows several panes which group the input values for the training dataset that is about to be imported.

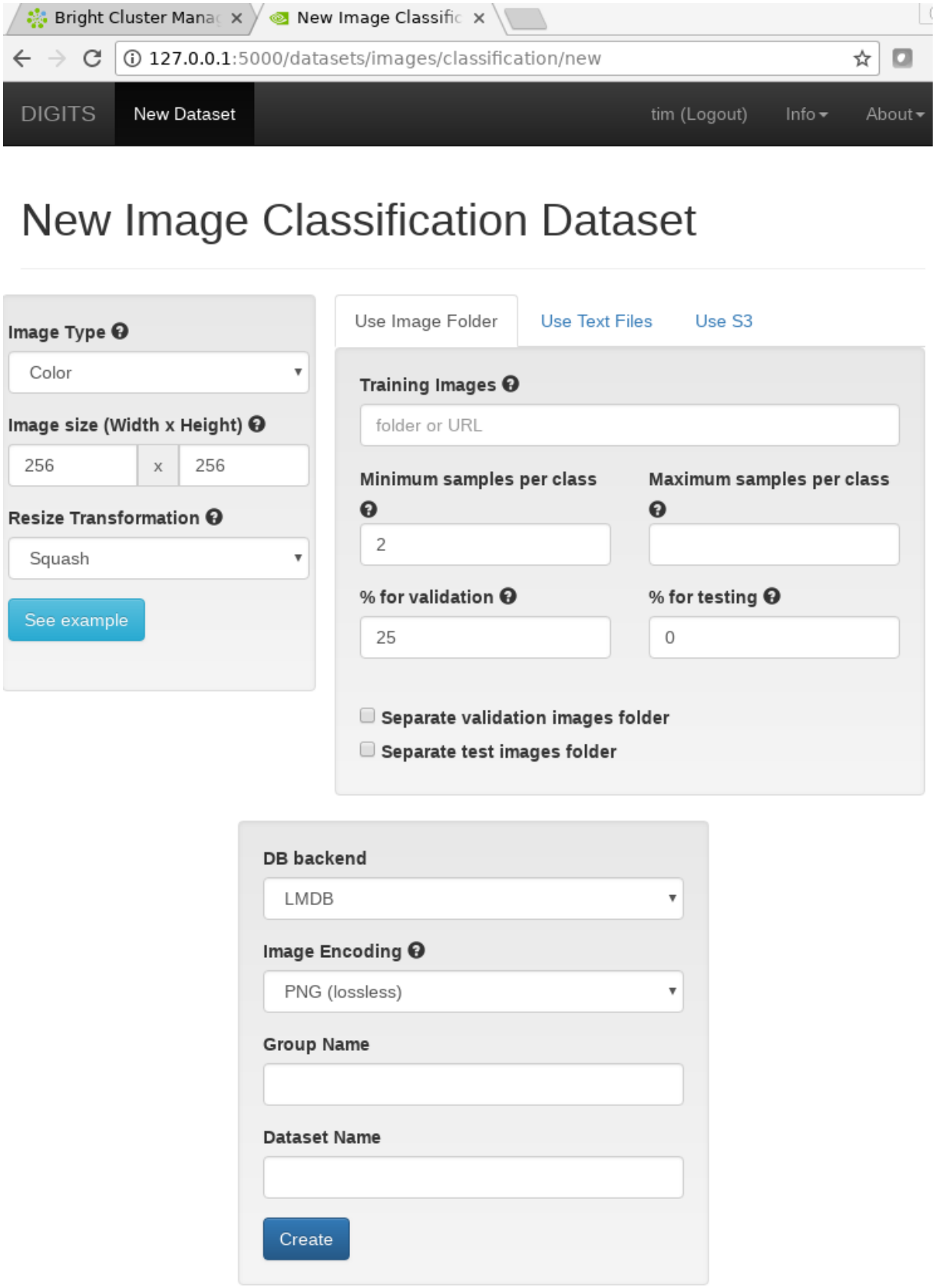

Figure 2.3: New Image Classification Dataset Dialog

To accept the training dataset:
For the pane starting with the field label `Image Type`:

- The `Image Type` should be set to `Grayscale`
- The `Image size` should be changed to 28 x 28

For the pane that has the tab label of `Use Image Folder`:

- For `Training Images`, the path to the MNIST training images should be entered. For the download described earlier in section 2.1 the path would be `/home/tim/train`.
- For the checkbox options:
  - The folder of MNIST test images, `/home/tim/test` in this session, can optionally also be added as a `Separate validation images folder`.
  - The `Separate test images folder` should not be used—test images are not used for anything in DIGITS yet.

For the pane starting with the field label `DB backend`:

- The dataset should be given a name, for example `mnistdataset`

The `Create` button can then be clicked to start the job run to create the classification databases.
A new screen showing the classification job status is then displayed. The screen is scrollable in the browser. The screen is shown in this manual, for printing reasons, as a top part (figure 2.4) and a bottom part (figure 2.5).
A `Job Status` pane at the top right hand side (figure 2.4) shows the expected job completion time if it is refreshed before the job completes, and shows the time the job took after it has completed.

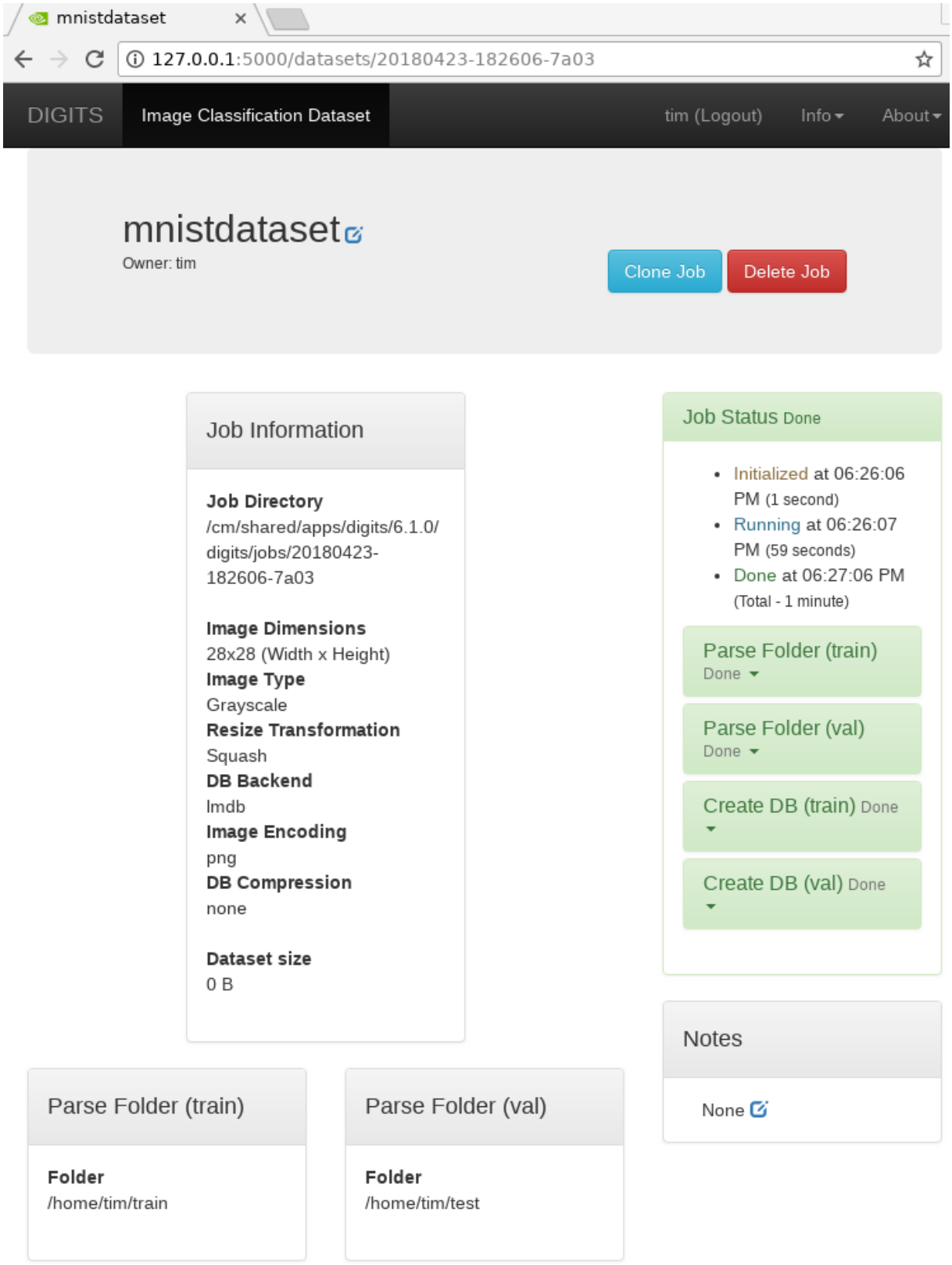

Figure 2.4: DIGITS MNIST Job Run Screen, Top Part

The bottom section (figure 2.5) allows access to the input and log files.

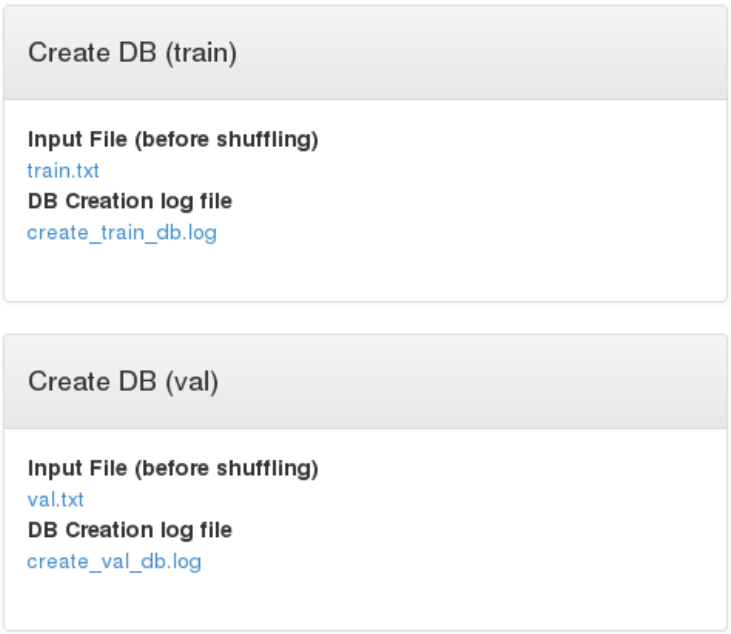

Figure 2.5: DIGITS MNIST Job Run Screen, Bottom Part

After the training and validation database import run is complete, the classification spread can be visualized in the refreshed updated bottom section (figure 2.6).

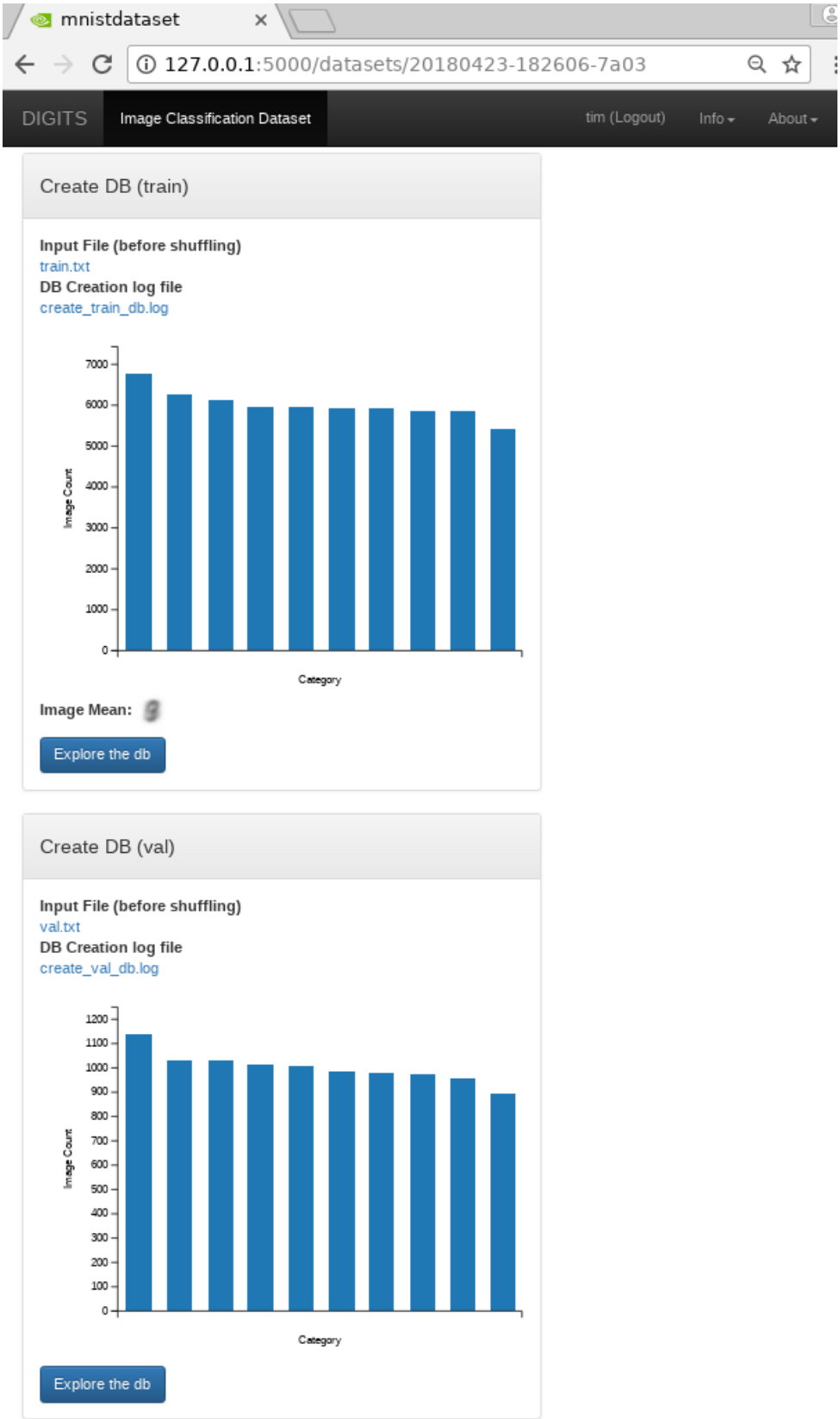

Figure 2.6: DIGITS MNIST Job Run Screen, Bottom Part, After Run

The corresponding training (`train`) and validation (`val`) database entries can be explored by click-

ing the Explore the db button.

Clicking DIGITS in the top left hand part of the web page brings up the home page again. Within the Datasets tab (figure 2.7), the name mnistdataset that was set earlier is now visible. Clicking on the name brings up a screen where like in figure 2.6 the classification spread can be visualized, and the corresponding training (train) and validation (val) database entries can be explored by clicking the associated Explore the db buttons.

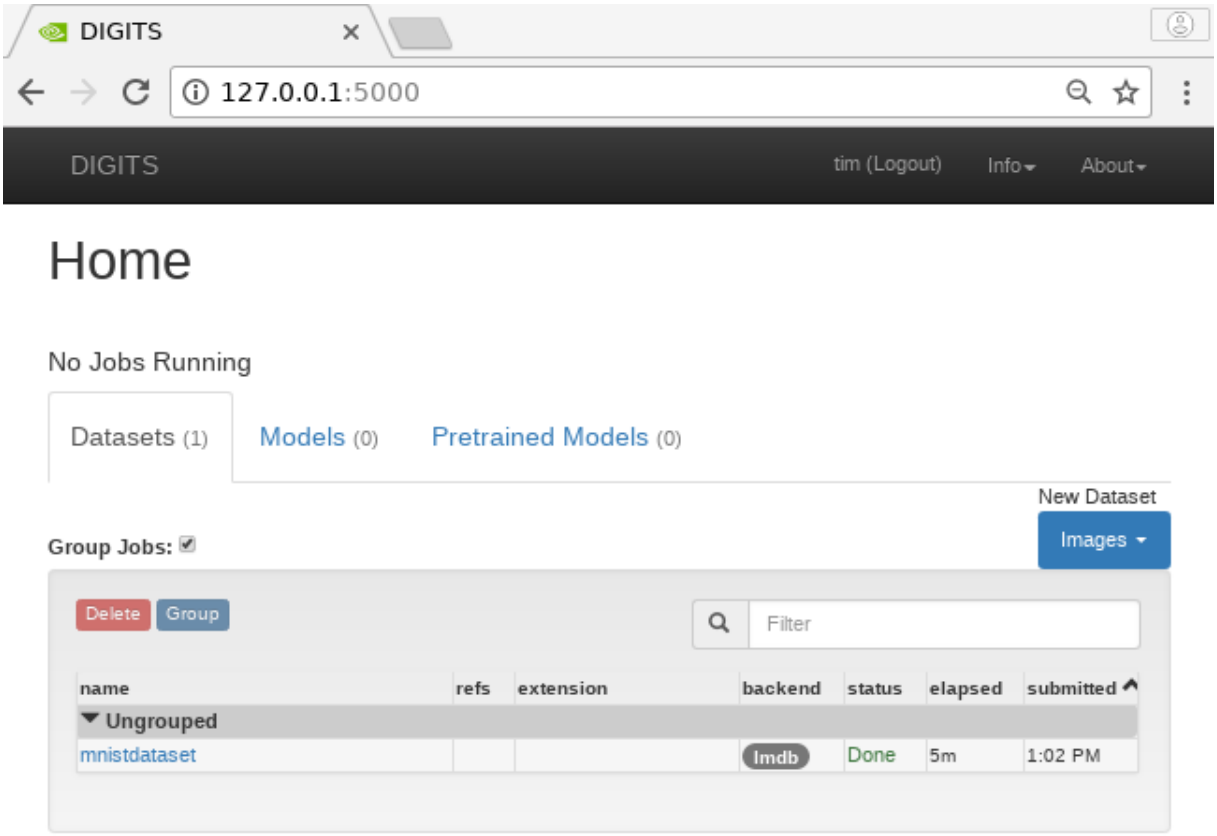

Figure 2.7: Generated Dataset Tab

2.7 Training A Model

The datasets can now be used to train the LeNet model.

Selecting the Models main tab of figure 2.7, then clicking on the Images button that is associated with the New Model label, and then clicking on the Classification option, opens up the New Image Classification Model page. The top part of the page is shown in figure 2.8:

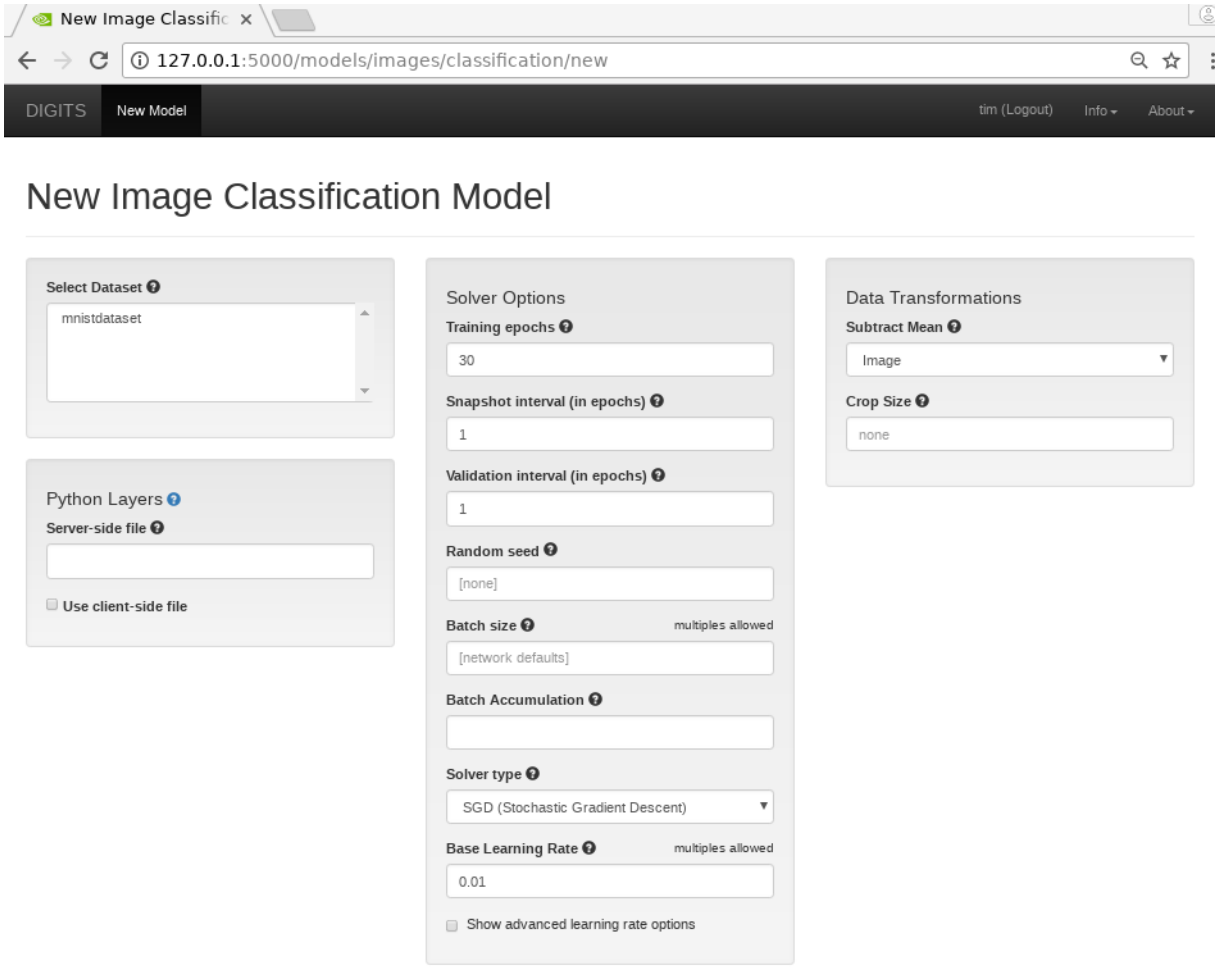

Figure 2.8: New Image Classification Model Top Part

The bottom part of the page (figure 2.9) shows network options.

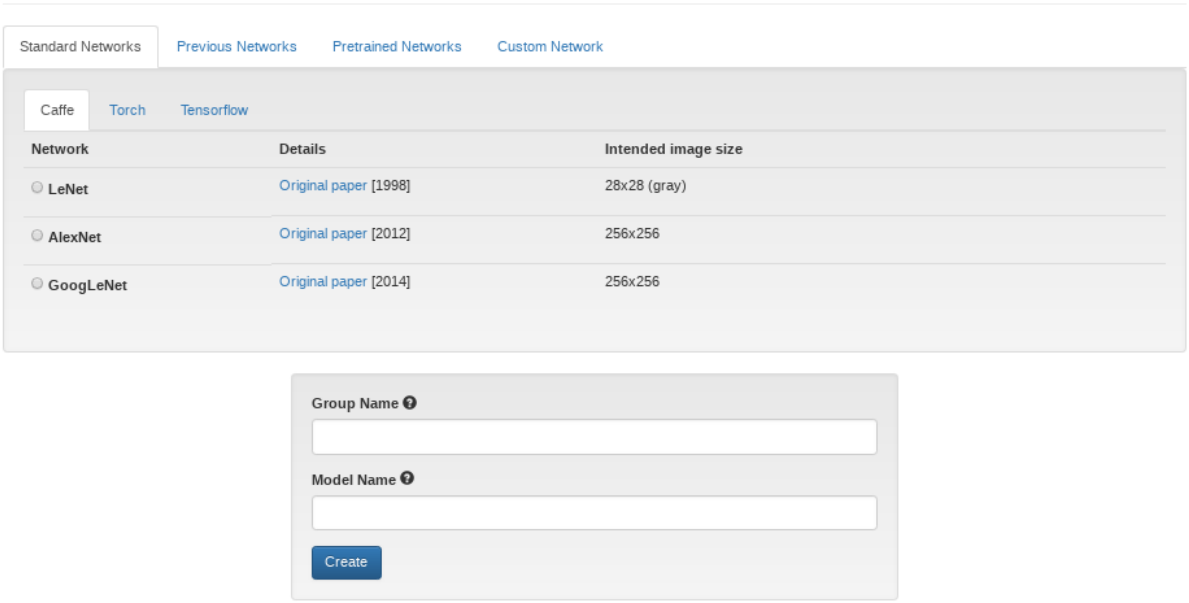

Figure 2.9: New Image Classification Model Dialog Bottom Part

In the `New Image Classification Model` page the following steps can be followed to classify the data:

- `mnistdataset` is selected in the `Select Dataset` field
- The `LeNet` network can be selected from the `Standard Networks` tab
- The model is given a name, for example `Len`
- The `Create` button is clicked

The browser should then continuously update as the data classification takes place. While training the model, the expected completion time is seen on the right side (figure 2.10):

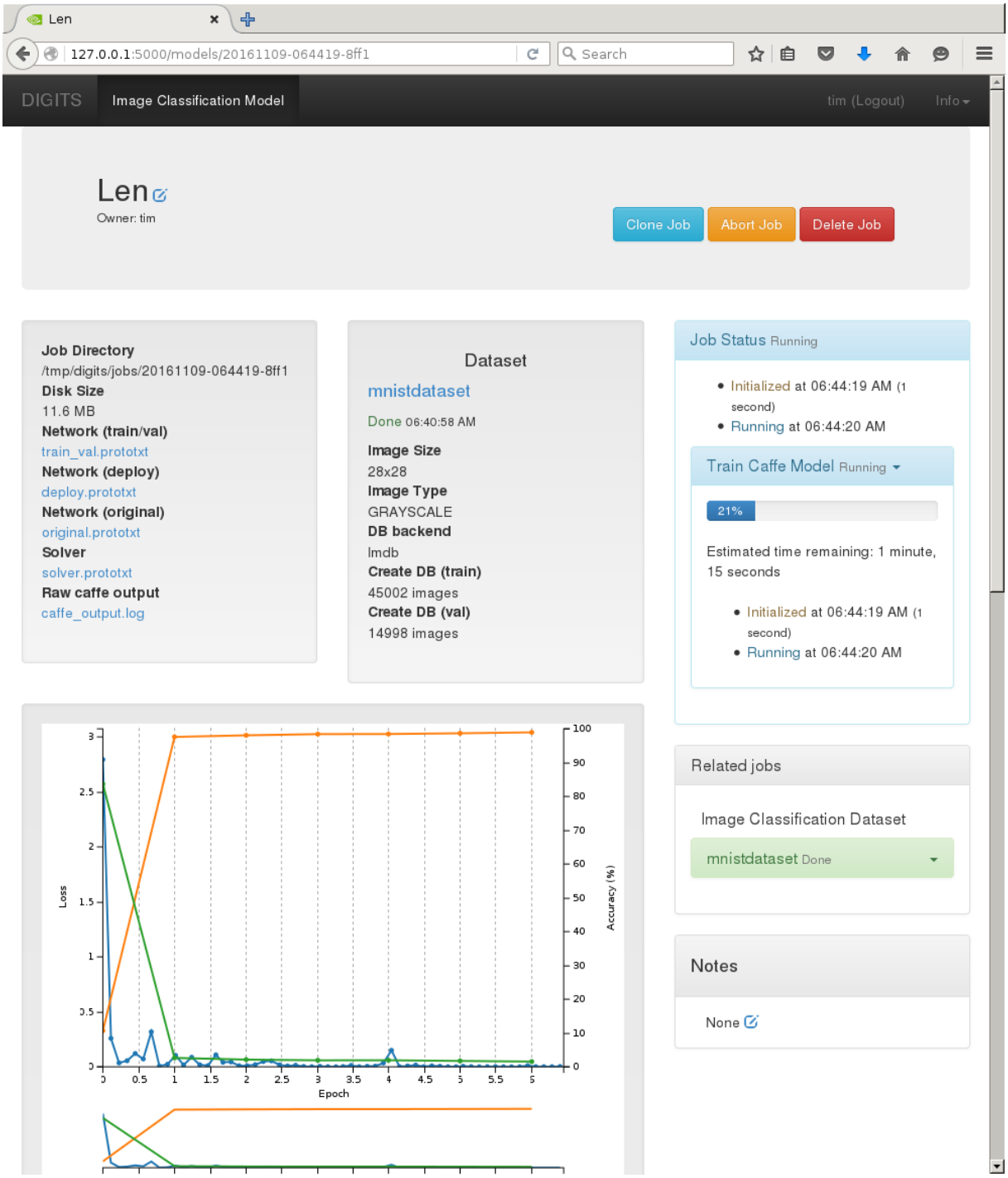

Figure 2.10: New Image Classification Model Training

2.8 Testing A Model

The bottom of the page (figure 2.11) allows the model to be tested.

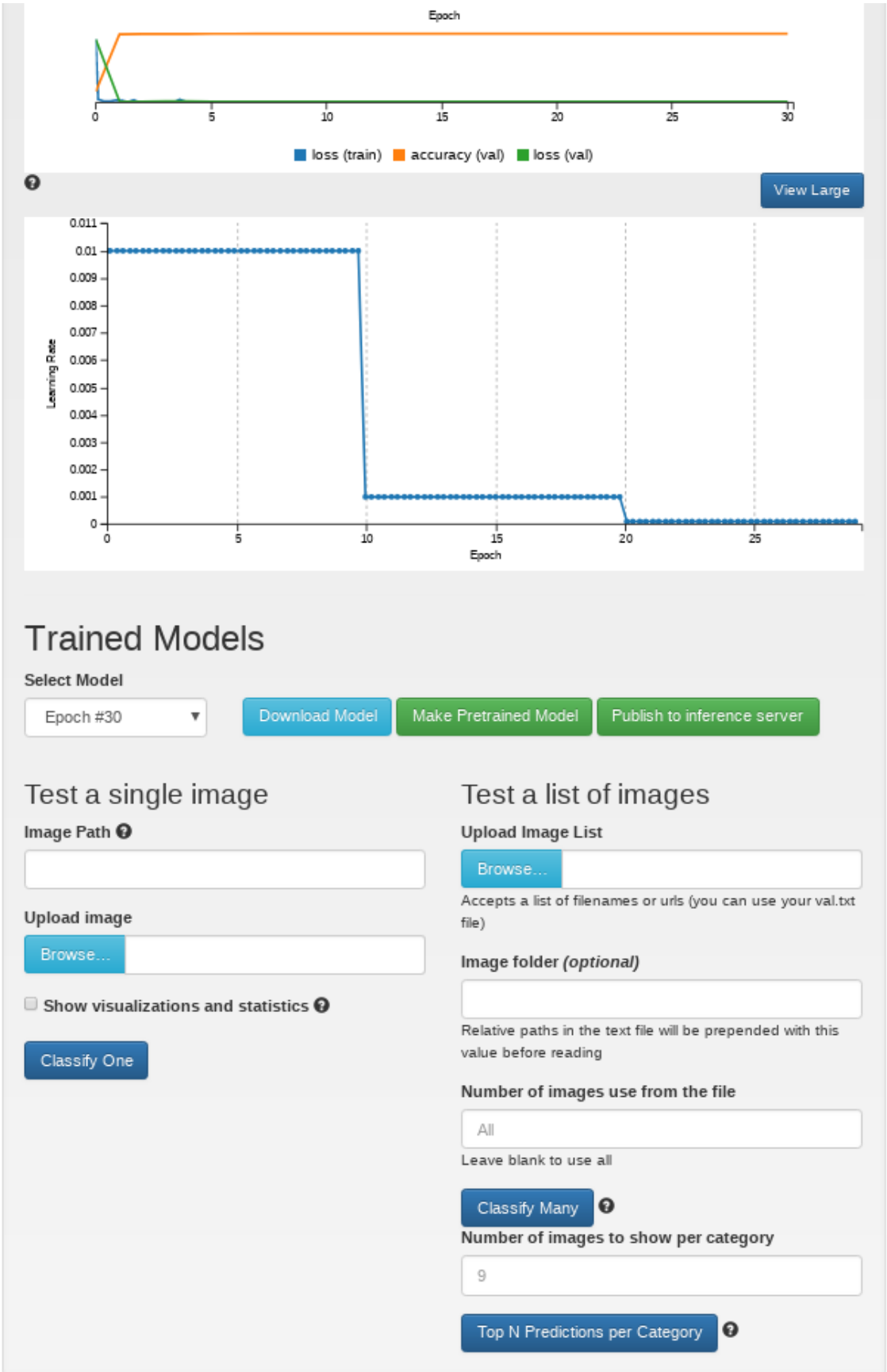

Figure 2.11: New Image Classification Of One Image

- The `Image Path` field, in the `Test a single image` column, can be used to select a file with a single image.
  - Images can be chosen from under the `/home/username/mnist/test/` directory on the server.
  - Alternatively an image from the web can have its URL pasted in the field.
- The `Upload Image` field can be used to upload an image. For example:
  - An Images can be uploaded from the local drive of the machine where the browser is running from.
  - An image of a digit can be created by hand in an image manipulation software such as Gimp.
- The `Show visualizations and statistics` box can be checked for extra information on how the algorithm was applied.

Clicking on the `Classify One` button then classifies that single image and displays results in a new window (figure 2.12):

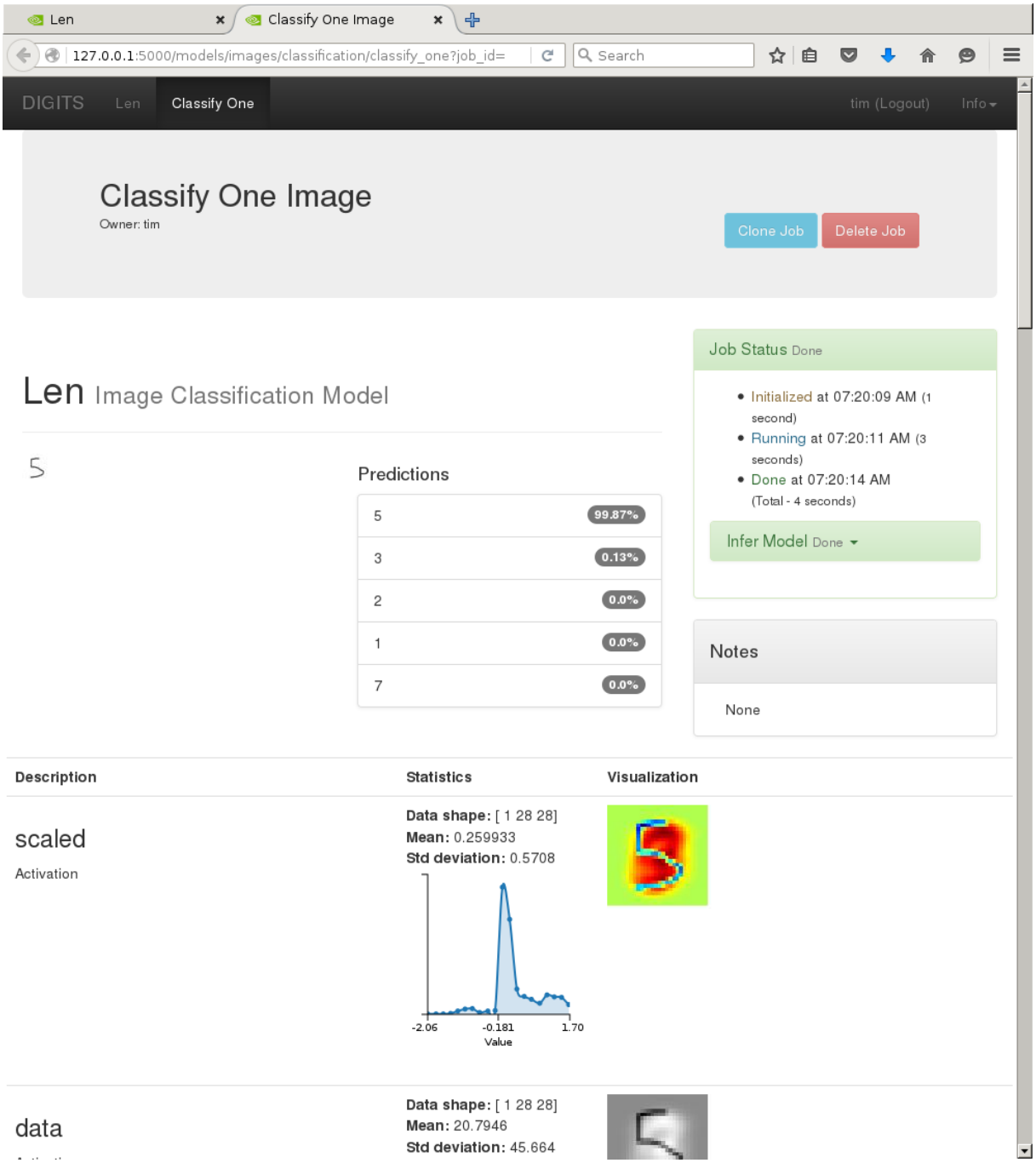

Figure 2.12: New Image Classification Result For One Image

At the top of the page, DIGITS displays the top five classifications and corresponding confidence values. DIGITS also provides further visualizations and statistics about the weights and activations of each layer of the network.

3

Running TensorFlow

This chapter goes through some example runs with TensorFlow. The INFO output messages have been removed in the runs for readability.

3.1 Hello World

A "`Hello World`" example that just shows that the software is in place for TensorFlow can be run as follows:

Example

```
[root@bright81 ~]# module load tensorflow
[root@bright81 ~]# module load openmpi/cuda/64
[root@bright81 ~]# python
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf

>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()

name: Tesla K40c major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:00:08.0
totalMemory: 11.17GiB freeMemory: 11.09GiB

>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

3.2 Training A Convolutional Neural Network

The following trains a convolutional neural network similar to LeNet-5, as explained in `https://www.tensorflow.org/versions/r0.11/tutorials/mnist/beginners/index.html`.

The code uses the TensorFlow convolutional module at `/cm/shared/apps/tensorflow/current/models/tutorials/image/mnist/convolutional.py`. It picks up training images and labels from the MNIST site, and places them in a directory `data` if it needs to. The images are then used to train and validate the model.

Example

```
[root@bright81 ~]# cd /cm/shared/apps/tensorflow/current/models/tutorials/image/mnist/
[root@bright81 mnist]# python convolutional.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz

2018-05-02 13:37:08.796152: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1344]\
 Found device 0 with properties:
name: Tesla K40c major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:00:08.0
totalMemory: 11.17GiB freeMemory: 11.09GiB
Initialized!

Step 0 (epoch 0.00), 25.9 ms
Minibatch loss: 8.334, learning rate: 0.010000
Minibatch error: 85.9%
Validation error: 84.6%
Step 100 (epoch 0.12), 158.8 ms
Minibatch loss: 3.267, learning rate: 0.010000
Minibatch error: 7.8%
Validation error: 7.5%
Step 200 (epoch 0.23), 154.9 ms
Minibatch loss: 3.355, learning rate: 0.010000
Minibatch error: 9.4%
...
Validation error: 0.8%
Step 8400 (epoch 9.77), 160.8 ms
Minibatch loss: 1.595, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Step 8500 (epoch 9.89), 157.6 ms
Minibatch loss: 1.611, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Test error: 0.7%
[root@bright81 ~]#
```

3.3 Image Recognition

The following session shows a pre-trained model running the `classify_image.py` script to recognize a test image of a panda. The image can be from anywhere, but the `wget` used in the session picks up a .tgz file that has the one used in the exercise at `https://www.tensorflow.org/tutorials/image_recognition`:

Example

```
[root@bright81 ~]# module load tensorflow
[root@bright81 ~]# cd /cm/shared/apps/tensorflow/current/models/tutorials/image/imagenet
[root@bright81 imagenet]# wget http://download.tensorflow.org/models/image/imagenet/\
inception-2015-12-05.tgz
```

```
...
Length: 88931400 (85M) [application/x-compressed-tar]
Saving to: 'inception-2015-12-05.tgz'

100%[======================================================>] 88,931,400  10.9MB/s  in 8.5s

2018-05-02 17:59:32 (9.99 MB/s) - 'inception-2015-12-05.tgz' saved [88931400/88931400]

[root@bright81 imagenet]# tar xvzf inception-2015-12-05.tgz
classify_image_graph_def.pb
cropped_panda.jpg
imagenet_2012_challenge_label_map_proto.pbtxt
imagenet_synset_to_human_label_map.txt
LICENSE
[root@bright81 imagenet]# python classify_image.py --image_file cropped_panda.jpg
2018-05-02 18:03:05.773697: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1344]\
 Found device 0 with properties:
name: Tesla K40c major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:00:08.0
totalMemory: 11.17GiB freeMemory: 11.09GiB
2018-05-02 18:03:06.725257: W tensorflow/core/framework/op_def_util.cc:343] Op Batch\
NormWithGlobalNormalization is deprecated. It will cease to work in GraphDef version\
 9. Use tf.nn.batch_normalization().
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89107)
indri, indris, Indri indri, Indri brevicaudatus (score = 0.00779)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00296)
custard apple (score = 0.00147)
earthstar (score = 0.00117)
[root@bright81 imagenet]#
```

3.4 Iris Classification

The iris flower data set is a standard raw data set, originally measured by R. A. Fisher in 1936. The data values in the set are a list of, per flower, the petal length and width, and the sepal length and width. Sepals are leaflike "under petals" under the actual petals. Based on these 4 attributes per flower, the fifth attribute—the species that the iris flower belongs to—can be determined. The determination of the species does not have an obvious solution because it depends on multiple variables at the same time. Fischer came up with a mathematical method to determine the species.

In machine learning, the iris flower data set is commonly used to illustrate concepts. An iris classification program, `premade_estimator`, is therefore a core TensorFlow program. The following session shows a run with `premade_estimator`, which downloads the data values used for training if required (some output ellipsized or removed for clarity):

```
[root@bright81 ]# cd /cm/shared/apps/tensorflow/current/models/samples/core/get_started
[root@bright81 get_started]# python premade_estimator.py
Downloading data from http://download.tensorflow.org/data/iris_training.csv
16384/2194 [==============================================================- 0s 0us/step
Downloading data from http://download.tensorflow.org/data/iris_test.csv
16384/573 [==================================================- 0s 0us/step
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmpLqap1u
INFO:tensorflow:Using config: '_save_checkpoints_secs': 600, ' ...'_save_summary_steps': 100
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
```

```
INFO:tensorflow:Graph was finalized.
2018-05-02 15:50:53.815952: I tensorflow/co...Found device 0 with properties:
name: Tesla K40c major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:00:08.0
totalMemory: 11.17GiB freeMemory: 11.09GiB
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1 into /tmp/tmpLqap1u/model.ckpt.
INFO:tensorflow:loss = 192.68596, step = 1
INFO:tensorflow:global_step/sec: 468.658
INFO:tensorflow:loss = 12.971608, step = 101 (0.214 sec)
INFO:tensorflow:global_step/sec: 685.834
...
INFO:tensorflow:loss = 4.189755, step = 901 (0.158 sec)
INFO:tensorflow:Saving checkpoints for 1000 into /tmp/tmpLqap1u/model.ckpt.
INFO:tensorflow:Loss for final step: 6.7716765.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2018-05-02-13:50:58
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpLqap1u/model.ckpt-1000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2018-05-02-13:50:58
INFO:tensorflow:Saving dict for global step 1000: accuracy = 0.96666664,...loss = 1.6617917

Test set accuracy: 0.967

INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpLqap1u/model.ckpt-1000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.

Prediction is "Setosa" (99.9%), expected "Setosa"

Prediction is "Versicolor" (99.5%), expected "Versicolor"

Prediction is "Virginica" (97.7%), expected "Virginica"
```

More information on the program can be found at `https://www.tensorflow.org/get_started/get_started_for_beginners`.

4

Jupyter And JupyterHub Usage

This chapter covers the usage of Jupyter and JupyterHub, as well as the possibility to integrate it alongside a Spark deployment within Bright Cluster Manager.

Jupyter on its own is single user. However JupyterHub allows it to provide a multi-user service, and is therefore commonly installed with Jupyter. In any case, in Bright Cluster Manager, the package `cm-jupyterhub` depends upon `cm-jupyter`, and so in this chapter, Jupyter and JupyterHub will be referred by only JupyterHub

To be able to use it as multi-user, it is necessary to have JupyterHub. It is possible to install and use Jupyter alone. The package `cm-jupyter` won't trigger the installation of `cm-jupyterhub`. However, this chapter covers the most common case, which is to have them working together. Indeed, the Bright Cluster Manager package `cm-jupyter` is a dependence of `cm-jupyterhub`. Therefore, in this chapter, the combination of Jupyter and JupyterHub are conveniently referred to as JupyterHub.

4.1 Installation Options

Bright Cluster Manager provides a `cm-jupyterhub-setup` script to install JupyterHub and integrate it with Apache Spark.

By default, Jupyter comes with kernels for `python2` and `python3`. `cm-jupyterhub-setup` can also generate additional kernels for the selected Spark instances. `cm-jupyterhub-setup` will generate at least 3 kernels, leveraging Apache Toree, for: Scala, PySpark, Spark SQL, and SparkR (only if `R` is installed). Only Apache Spark versions starting from 2.0.0 are supported.

`cm-jupyterhub-setup` allows JupyterHub to be independently deployed, and generates Spark kernels. Apache Spark must be already installed (Chapter 5 of the *Big Data Deployment Manual*).

If Apache Spark is already deployed, users can select the option `Deploy` and then choose the desired Spark instance. If Apache Spark is deployed after JupyterHub installation, users can re-run `cm-jupyterhub-setup` in order to generate additional kernels, by selecting the option `Integrate`.

`cm-jupyterhub-setup` allows users to set some options via `Advanced module configuration`:

- `c.JupyterHub.port` for the proxy port (default: 8000)
- `c.JupyterHub.hub_port` for the hub port (default: 8082)
- `c.Spawner.env_keep` for the environment variables to pass to the spawned process
- `User for service` for the user to use for running the service (default: `root`)

`cm-jupyterhub-setup` will set up the proxy to use HTTPS on the selected port, using a self-signed certificate. By default, the `cm-jupyterhub` service will run as `root` user. The wizard will show a list of users (retrieved via LDAP) to utilize as alternative. This technique is implemented via the `SystemdSpawner` class for JupyterHub and some changes in the sudoers configuration. Specifically, CMDaemon will create the file `/etc/sudoers.d/jupyterhub` in order to allow the selected user to spawn processes as the user logged in via the hub.

4.1.1 Verifying Jupyter And JupyterHub Installation

After the `cm-jupyterhub` service is started, it can take some time until the service is fully up and running. Even if `systemctl status cm-jupyterhub -l` shows that the service is already running, it can still take some seconds longer start functioning.

At this point, whatever the method that was used to install JupyterHub, the `cm-jupyterhub` service should be running on the nodes. Each node should be accessible via a web browser on port `8000` (figure 4.1):

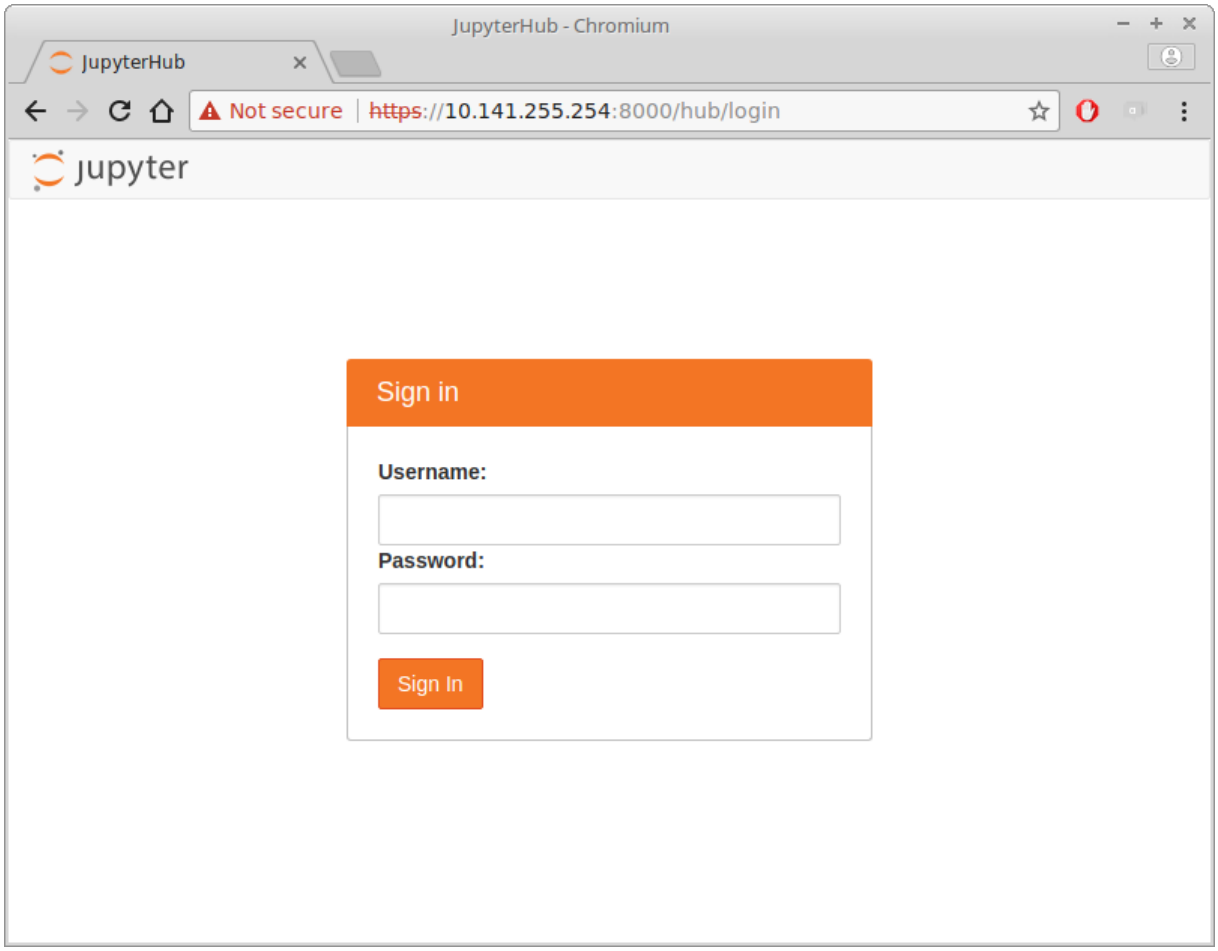

Figure 4.1: JupyterHub login screen

4.2 Creating And Running A Notebook

Any user registered in the Linux-PAM system can log in to JupyterHub. For example, a test user `jupyterhubuser` with password `pw1` can be created with:

Example

```
[root@bright81 ~]# cmsh -c "user ; add jupyteruser ; set password pw1 ; commit "
```

To be able to access the HDFS in the case of integrated deployment with Spark Yarn, this user has to be granted access. Access can be granted with the `cm-hadoop-user` utility, for a given instance `hdfs1`, with:

Example

```
[root@bright81 ~]# cd /cm/local/apps/cluster-tools/hadoop
[root@bright81 hadoop]# cm-hadoop-user --grant jupyteruser=hdfs1
```

A login can then be carried out for the test user in the JupyterHub screen. The user is forwarded to a Jupyter instance (figure 4.2):

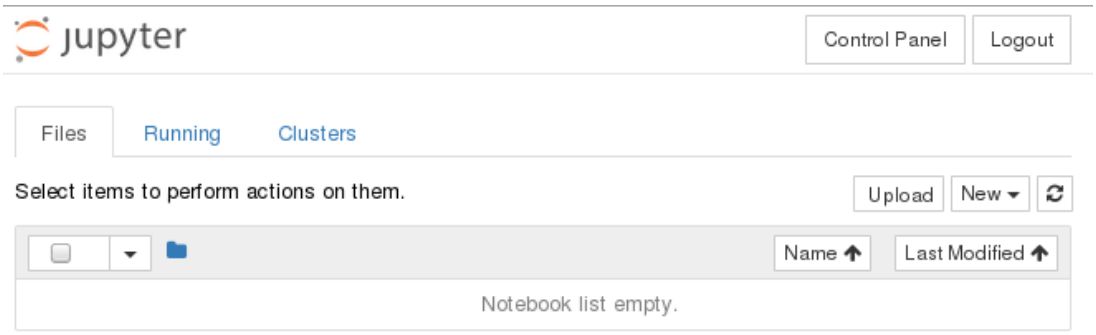

Figure 4.2: JupyterHub landing screen

Clicking on the `New` button of figure 4.2 displays a list of kernels (figure 4.3):

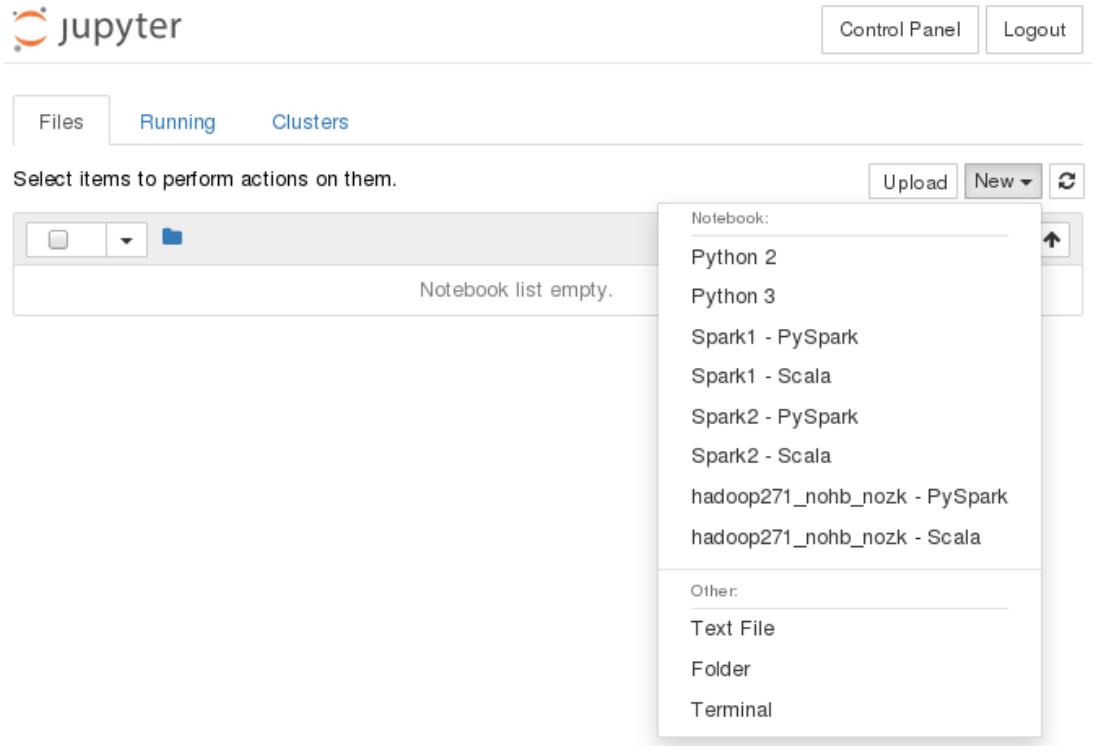

Figure 4.3: JupyterHub kernel list

The two first kernels in figure 4.3, `Python 2` and `Python 3`, are default kernels. At least one of them will be present.

The remaining ones in the example were installed using the integration methods.

If `Python 3` is chosen, then a notebook can be created for it (figure 4.4):

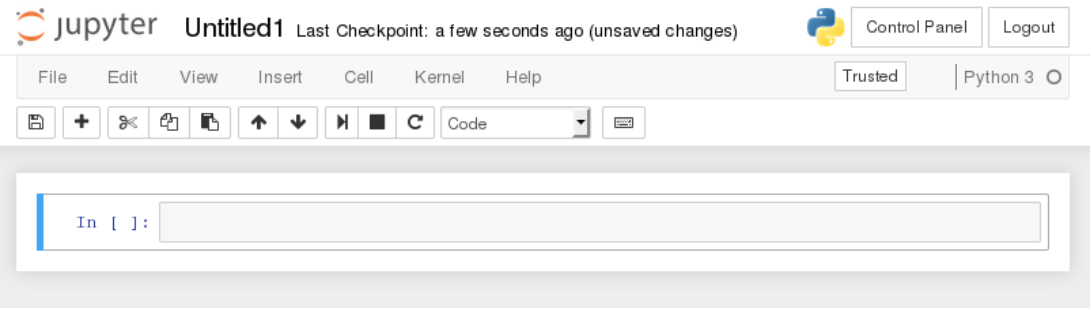

Figure 4.4: Sample Python 3 notebook

A simple Python 3 code, such as `print('Hello')` can be typed in the text entry box and run (figure 4.5):

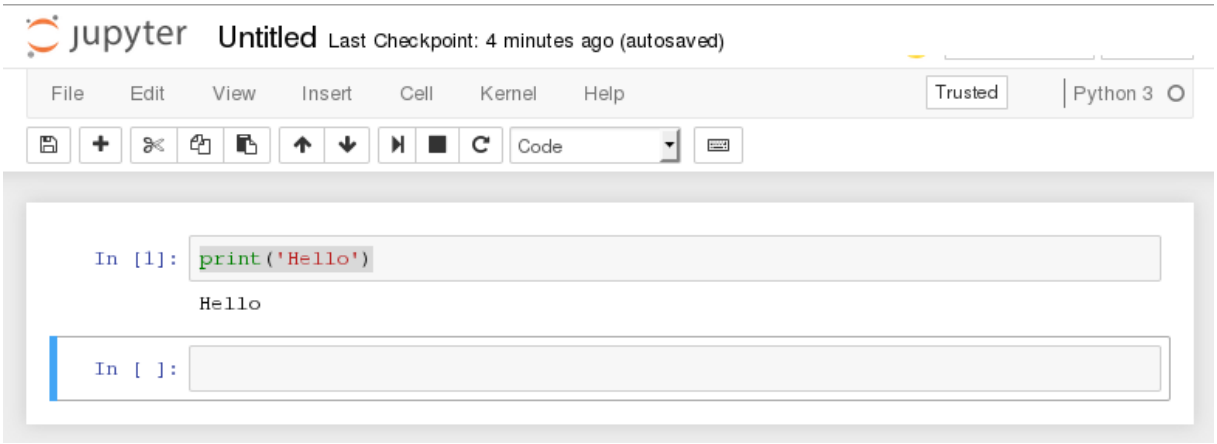

Figure 4.5: Sample Python 3 notebook: Run

It is not possible to import `pyspark` here, because the kernel does not know about the Spark deployment location:

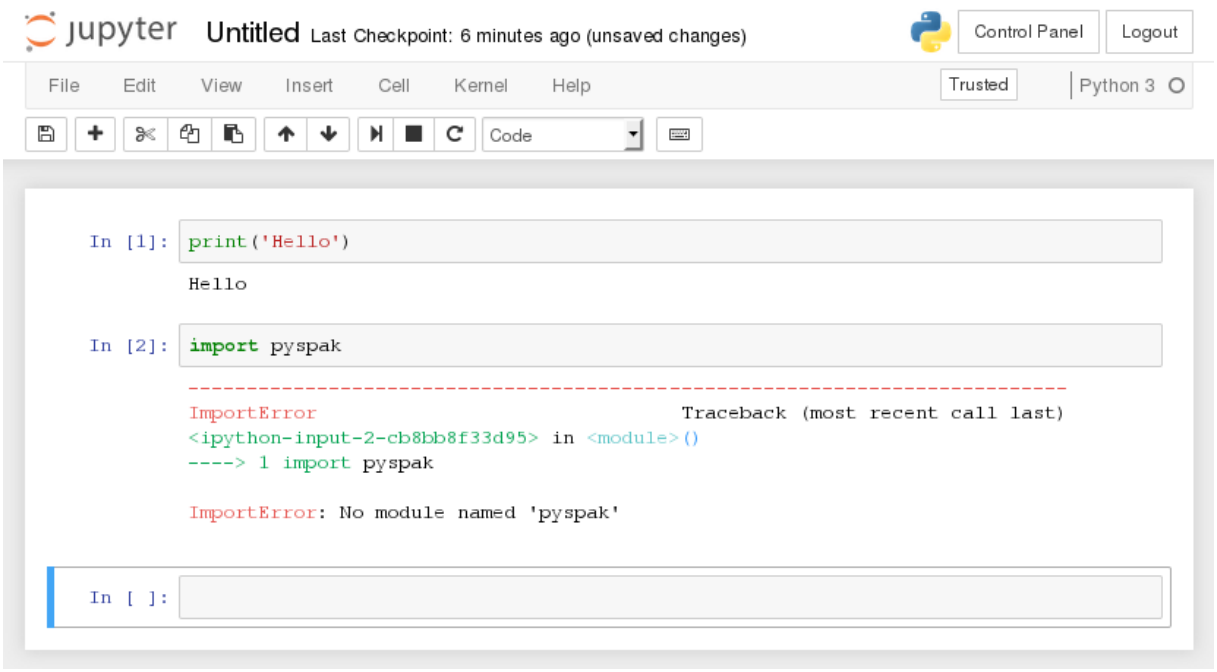

Figure 4.6: Sample Python 3 notebook: PySpark import not possible

The notebook can be closed by clicking on `File` and `Close and Halt`.

If the user has any kernel for a Spark deployment, then a notebook can be created for it. In this case, no error is shown after importing `pyspark`, or even accessing the Spark context automatically created by Spark directly (figure 4.7):

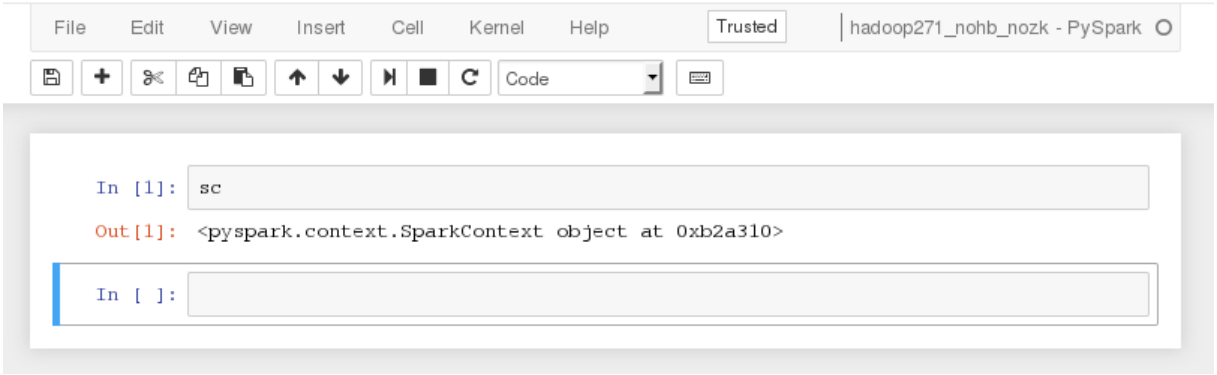

Figure 4.7: Sample integrated Spark notebook: successful import PySpark/access Spark context

That same context is available in all Toree kernels, such as the Scala one:

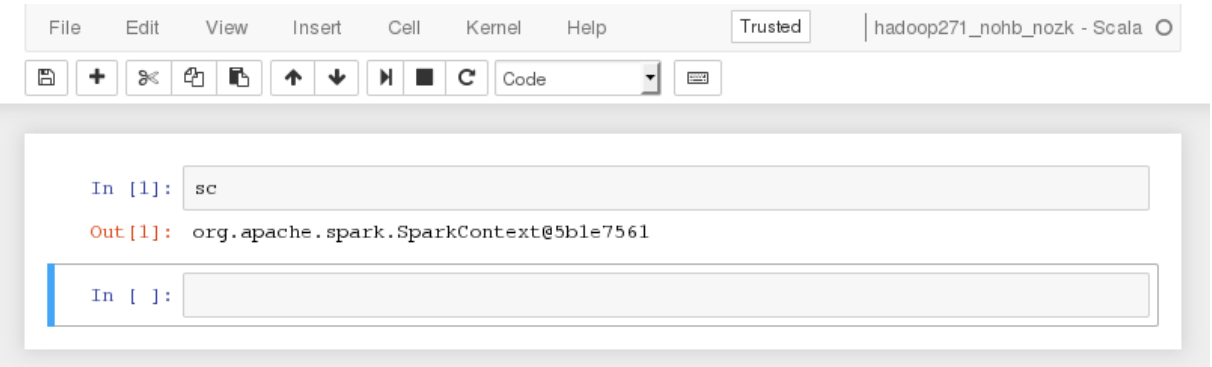

Figure 4.8: Sample integrated Spark notebook: successful access Spark context with Scala

Extra caution is required with the `SQL` Toree kernel, as of the writing of this chapter. The kernel hangs if given invalid SQL queries. This is an open issue at `https://issues.apache.org/jira/browse/TOREE-419` at the time of writing of this section (May 2018).

4.3 An Example Of A Notebook Connecting To Spark: Word2Vec

If JupyterHub deployment has been carried out as in the preceding sections, and it has been verified and its notebook creation capability checked, then it should be ready for use. An example run is now carried out in this section.

The machine learning library code `Word2Vec` at `https://spark.apache.org/docs/latest/mllib-feature-extraction.html#word2vec` takes a dataset of words, and outputs a set of words that can be used in similar contexts. All that is needed is to place the sample data file in a location that both `jupyteruser` and Spark users can access.

To run the example, a Python programming language, and a Spark standalone deployment, are chosen. The kernel PySpark for Spark standalone is an appropriate choice. The preparation can be carried out as follows:

Example

```
[root@bright81 ~]# su - jupyteruser
[root@bright81 ~]# wget http://mattmahoney.net/dc/text8.zip
[root@bright81 ~]# unzip text8.zip
[root@bright81 ~]# truncate  -s 10000000 text8  # truncate to 9.6 MB file
[root@bright81 ~]# sudo su - spark ls $PWD/text8 # ensure spark user can access it
[root@bright81 ~]#
```

The `truncate` step ensures the file is small enough to run in a cluster with few resources. Truncation can be skipped for clusters that can run large enough instances.

Instead of PySpark and Spark standalone, Scala and Spark YARN could have been chosen instead. In this case, the file should be uploaded to the HDFS. To use a bare Python 2 or Python 3 kernel, the user must also take care to obtaining the context `sc`, which is set automatically by the integrated kernels.

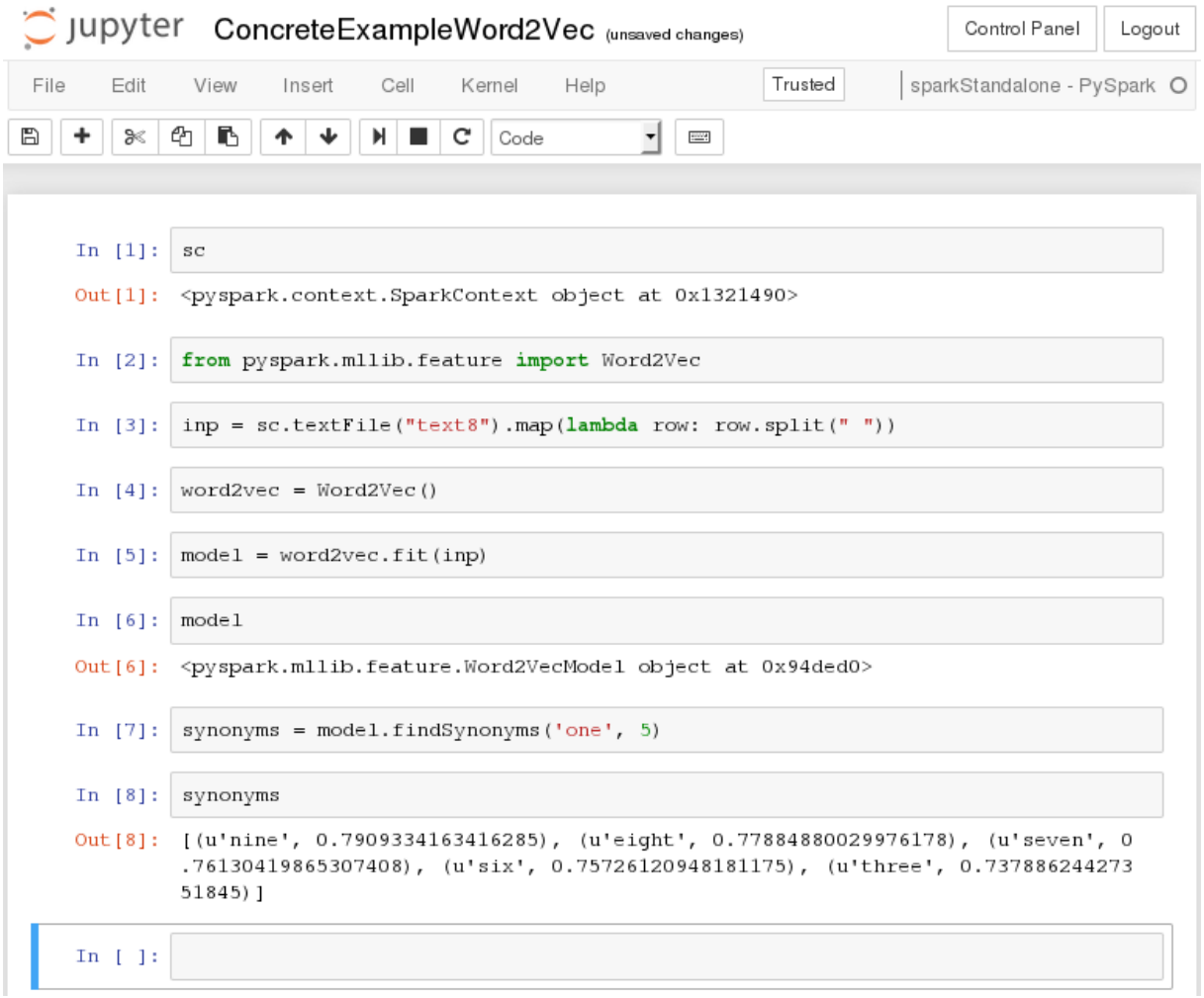

Figure 4.9: Word2Vec Example

Cell 1 shows that the `sc` context is properly set. The Spark RDD is created in cell 3 using the source file as input.

Cell 2 shows the Word2Vec model factory being imported. It takes the Spark RDD to create Word2VecModel in cell 5. This is where intensive computation is carried out, and the user may expect significant latency, as indicated by an asterisk `[*]`, depending on the cluster and RDD size. If no errors as cell 2 to 5 are processed, then no output is expected.

Cell 6 shows the output of `model`, and ensures all is well.

In cell 7, the already-computed model is queried to fetch 5 synonyms for the word `one`. Synonyms in the context of Word2Vec can differ from the concept as understood by humans.

In cell 8 the synonym output is shown, along with correlation coefficients. The synonyms are `nine`, `eight`, `seven`, `six`, and `three`.

4.4 Removal Of JupyterHub

Before removing JupyterHub, the administrator should ensure that all kernels have been halted and that no user is still logged onto `cm-jupyterhub`. Stopping `cm-jupyterhub` services with users that are still logged in, or with running kernels, has undefined behavior.

When removing a Spark instance, the corresponding Jupyter kernels will be removed, but not JupyterHub itself, as in the following example.

Example

```
[root@bright81 ~]# cm-spark-setup -u hdfs1
Undoing Jupyter, JupyterHub and Toree integration... done.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Cleaning ZooKeeper... done.
Removing additional Spark directories... done.
Removing Spark-related metrics... done.
Removal successfully completed.
Finished.
```

Conversely, in order to remove JupyterHub, the script `cm-jupyterhub-setup` must be run, either in interactive mode, or with the option `--remove`.

In any case, for a complete cleanup, the following packages must be removed: `cm-jupyterhub`, `cm-jupyter`, `cm-npm-configurable-http-proxy`.

5

Running PyTorch

5.1 Introduction

Torch is a machine learning library that was originally used with the Lua language as a scripting front end.

PyTorch allows the popular Python language to be used as a front end with Torch instead. PyTorch is designed as a replacement for NumPy when GPU computation is to be used. For GPU computation, PyTorch uses multidimensional arrays called tensors[1].

At the time of writing (May 2018) PyTorch is under heavy development. It is expected to make its way to version 1 soon, but is already in use at some very large companies such as Facebook.

The publicly available version of PyTorch at the time of writing is version 0.4, which has NumPy support removed for Python3, due to bugs.

5.2 Using Tensors

Appropriate environment modules should first be loaded. Because of the heavy development of PyTorch at the time of writing (May 2018), and special requirements, there may be changes in what actually works for a PyTorch user. In general, Python3 versions of the modules are recommended, along with Python version 3.6 as the recommended Python 3 version.

A module stack that works with the PyTorch tensor examples described in this section at the time of writing is:

Example

```
[root@bright81 ~]# module list
Currently Loaded Modulefiles:
 1) shared                    6) nccl2/2.1.15           11) ml-python3/pytorch/0.4.0a0
 2) cmsh                      7) torch7/7.0             12) cuda91/toolkit/9.1.85
 3) cmd                       8) gcc5/5.5.0             13) openblas/dynamic/0.2.20
 4) cluster-tools/8.1         9) hpcx/2.0.0
 5) cm-ml-python3deps/2.0.0  10) openmpi/cuda/64/3.0.0
```

Many users may be familiar with NumPy, an older Python library for numerical calculations.

As an introduction to PyTorch, a non-PyTorch way with Numpy is used to set up random arrays and carry out a simple two layer network calculation as described in `https://pytorch.org/tutorials/beginner/pytorch_with_examples.html#warm-up-numpy`. The file `https://pytorch.org/tutorials/_downloads/two_layer_net_numpy.py` can then be run.

Example

[1] Physicists and mathematicians may need to be aware that tensors in machine learning differ from (are a superset of) the archetypal tensors which they are familiar with and which obey coordinate system transformation rules. The tools and notation used in the analysis of machine learning tensors are however often borrowed from those used for archetypal tensors.

Example

```
[root@bright81 ~]# python3.6 two_layer_net_numpy.py
0 35041206.18035958
1 33697170.26473527
2 36394414.73114709
...
```

A PyTorch way of carrying out the same task, using tensors, and without using Numpy, is described in https://pytorch.org/tutorials/beginner/pytorch_with_examples.html#pytorch-tensors. For this, the file https://pytorch.org/tutorials/_downloads/two_layer_net_tensor.py is run.

Example

```
root@bright81 ~]# python3.6 two_layer_net_tensor.py
0 27050888.0
1 21890864.0
2 19408946.0
...
```

The autograd package in PyTorch provides functions to calculate tensor gradients. This is described in https://pytorch.org/tutorials/beginner/pytorch_with_examples.html#autograd, along with a code example that uses autograd. The code can be picked up from https://pytorch.org/tutorials/beginner/examples_autograd/two_layer_net_autograd.html#sphx-glr-beginner-examples-autograd-two-layer-net-autograd-py.

Example

```
root@bright81 ~]# python3.6 two_layer_net_autograd.py
0 37842080.0
1 35130852.0
2 34667068.0
...
```

More documentation and other PyTorch deep learning code samples and functions can be found at the https://pytorch.org website.